In the Specification

Please delete paragraph 0038 in its entirety and replace it with the following:

[0038] The data access layer 110 comprises a query processor 112, a metadata database 114, a transaction database 116, and an exception handling module 118. The query processor 112 receives data requests from the external application 120. A data request might be a query to retrieve data from a COTS data store 130, 140, or 150, an update to the data in a COTS data store 130, 140, or 150, or some other type of interaction with the data in a COTS data store 130, 140, or 150.

Please delete paragraph 0039 in its entirety and replace it with the following:

[0039] The metadata database 114 informs the query processor 112 how to handle data requests. A logical data model in the metadata database 114 determines which COTS data store 130, 140, or 150 a data request applies to, how the data should be updated, retrieved, aggregated, or otherwise manipulated and whether updated data should be sent to a data warehouse 160 or 170.

Please delete paragraph 0043 in its entirety and replace it with the following:

[0043] An external application 120 sends the data request to the query processor 112 via path 105. Query processor 112 consults the metadata database 114 via path 115 to determine how the data request should be handled. Query processor 112 also sends a copy of the data request to transaction database 116 via path 125. The logical data model in the metadata database 114 indicates to the query processor 112, via path 115, how to process the data request. The query processor 112 then attempts to

perform the appropriate actions on the COTS data stores 130, 140, and 150, via paths 135, 145, and 155, respectively.

Please delete paragraph 0044 in its entirety and replace it with the following:

[0044] In the embodiment of Figure 3, the data request is completed successfully in all of the COTS data stores 130, 140, and 150. Therefore, the copy of the data request in the transaction database 116 is no longer needed and is removed from the transaction database 116. The data request is then considered committed in the COTS data stores 130, 140, and 150 and the query processor 112 sends a report of a successful data request to the external application 120 via path 165. If the data request was a data update and if the logical data model in the metadata database 114 specifies that the updated data should be sent to the data warehouses 160 and 170, this is then done in a process described below.

Please delete paragraph 0045 in its entirety and replace it with the following:

[0045] The same data element can be maintained in more than one COTS data store 130, 140, and/or 150 and might also be kept in one or more data warehouses 160 and/or 170. For example, a common data element such as a user name might need to exist in multiple locations. If a change is made to such a data element, the change would typically need to take place in all locations that contain the element. For example, the metadata database [[14]] 114 might tell the query processor 112 that a data request needs to go to the COTS data stores 130, 140, and 150 in that order. When the change is committed in the COTS data stores 130, 140, and 150, the

metadata database [[14]] 114 might specify that the same change needs to occur in one or more data warehouses 160 and/or 170. While data requests to the COTS data stores 130, 140, and 150 are done synchronously, updated data may be sent to the data warehouses 160 and 170 in an asynchronous, publish/subscribe manner.

Please delete paragraph 0047 in its entirety and replace it with the following:

[0047] In the embodiment of Figure 3, the query processor 112, via paths 165 and 175, publishes any updates that were made to the COTS data stores 130, 140, and/or 150. A data warehouse 160 or 170 can subscribe to the updates, via path 165 or 175, if it is interested. If a data warehouse 160 or 170 subscribes to the updates, the data in the data warehouse 160 or 170 will match the data in the COTS data stores 130, 140, and 150.

Please delete paragraph 0048 in its entirety and replace it with the following:

[0048] In an embodiment, the original data request is not deleted from the transaction database 116 upon the successful updating of COTS data stores 130, 140, and 150 but instead remains in the transaction database 116 until a data warehouse 160 or 170 responds that it has received the updated data. This allows the query processor 112 to retrieve a data request from the transaction database 116 and publish a data update again if an error occurs in the publish/subscribe process.

Please delete paragraph 0049 in its entirety and replace it with the following:

[0049] In this manner, the data warehouses 160 and 170 can replicate the data in the COTS data stores 130, 140, and 150. After the data in the COTS data stores 130, 140, and 150 has been copied to the data warehouses 160 and 170 one time in a batch process, the data warehouses 160 and 170 can stay up-to-date merely by keeping up with changes in the COTS data stores 130, 140, and 150 as they occur. Updating a data warehouse 160 or 170 in this manner keeps the data warehouse 160 or 170 more current than copying data to the data warehouse 160 or 170 in a periodic batch process.

Please delete paragraph 0050 in its entirety and replace it with the following:

[0050] Figure 4 depicts the processing of a data request in which an error occurs. When an error occurs, the data access layer 110 allows the COTS data stores 130, 140, and 150 to be rolled back to their previous states. In the embodiment of Figure 4, an external application 120 makes a data request to COTS data stores 130, 140, and 150 sequentially as described above. In this case, however, an error occurs in the processing of the request by the Oracle database 140 after the request is successfully completed in the DB2 database 130. Therefore, the DB2 database 130 and the Oracle database 140 need to be in the status they held prior to the data request.

Please delete paragraph 0052 in its entirety and replace it with the following:

[0052] When an error occurs and the query processor 112 is unable to complete a data request, the query processor 112 informs the exception handling module 118 of the

error via path 185 and specifies the COTS data store 130, 140, or 150 in which the error occurred. The exception handling module 118 then retrieves the record of the data request from the transaction database 116 via path 195. In the embodiment of Figure 4, the exception handling module 118 retrieves the record of the data request made to the DB2 database [[30]] 130 and uses it to determine the prior state of the data in the DB2 database 130. Then, via path 205, the exception handling module 118 rolls the data in the DB2 database 130 back to its previous state.

Please delete paragraph 0053 in its entirety and replace it with the following:

[0053] The exception handling module 118 then informs the query processor 112, via path 185, of the actions it took in rolling back the data. The query processor 112 then sends an exception report to the requesting external application 120 via path 225 and removes the copy of the data request from the transaction database 116 via path 125.

Please delete paragraph 0054 in its entirety and replace it with the following:

[0054] In addition to the above functions, the query processor 112 can also perform security functions by calling an external security service 180, via path 235, to determine if the external application 120 is authorized to make a data request. The externalizing of security functions allows the use of a single sign-on for the data access layer 110 and for other applications in an enterprise.